WIRELESS PARKING SYSTEM

G52GRP YEAR 2 GROUP PROJECT

School of Computer Science, University of Nottingham Malaysia Campus

Group 2:
Lim Zhi En (khcy2lzn)
Tang Wei Qi (khcy2twi)
Michael Muita Mugo (kecy2mmg)
Mohammad Syamil (khcy2msr)

Supervisor:
Kr Selvaraj

4/21/2014

https://github.com/jakebooyah/wireless-parking-sys/

CONTENTS

1 INTRODUCTION

1.1 PROJECT BACKGROUND

Motor vehicles play a conspicuous role in the modern human life. Transport (especially cars) offers rapid, reliable and convenient mobility on demand to an ever-growing number of people in countries throughout the world. But for all their positives, automobiles carry with them many negatives. One of which is that it's difficult to find car park especially in public place such as colleges and universities. So do students and staff at the University of Nottingham Malaysia Campus. This results in most students being late for classes. This problem became more complicated with the recent changes in the traffic policy. With this problem in mind, there are a lot of solutions implemented to derive from, that help to overcome this problem. We can observe such measures being implemented in public places such as shopping malls.

Tracing along the same line of thought from existing solutions, we decided to develop an android application with the aim of assisting in finding free parking slots easily. Secondary research was carried out about existing systems in the market. This led to the conclusion that the existing system can be improved. We figured out that the existing system is not effective enough, as they just tell the drivers how many empty parking slots are left. This help users a lot but, but is still insufficient because the user does not know where exactly the empty car park is located. They still need to go around the car parks and randomly search where the free parking sis located. This leads to higher fuel consumption than needed.

With this in mind, the system to be developed shall help users locate where the available parking bays are, and to also save its parking location for easier location by the driver.

1.2 LITERATURE REVIEW

Parking management systems have become a crucial part to busy places whose parking lots are frequented a lot by drivers, especially with the large number of cars that are owned by people. The systems that have been put in place have helped drivers remember the location that they parked at, as well as provide them with real time updates to their availability. Based on our research on existing software systems, we found out that they offer various benefits. For example:

- **My Car Parking by DRD**

This is a free application on the android platform which allows users to:

- Save their location
- Provide guidance to the car location using Google Navigator based on Google Maps service
- Allows users to set a reminder for a parking ticket expiration date, with a notification alert about it

- **Sunway Pyramid Shopping Mall Car Park**

This is an integrated software system, which performs the following:

- Use billboard indicators to inform and guide drivers to available parking bays
- Use of infrared and ultrasound sensors to detect cars
- Display the number of parking bays that are available in a particular zone on sign boards
- Make use of real-time detection of parking bays and their availability

We saw that these benefits could also be applied to the University of Nottingham Malaysia Campus, and hence form the basis of this project.

Any driver that frequents the University of Nottingham Malaysia Campus knows of the pain and hassle often involved in finding the ideal parking spot for the day, and it is often a case of the early bird catches the worm scenario. As students part of this problem, we sought out to better manage frustrated drivers at the university, based on the previous research.

However, our implementation is unique in that it utilises proximity to buildings within the university, and provides with convenience-based querying of parking locations. Additionally, the implementation of sensors is in a more cost effective and eco-friendly solution which makes use of old Android smartphones.

This project set out to create a wireless parking mobile system, which would save the drivers at the University of Nottingham loads of time often spent in not only finding a parking spot that is available but one that is also convenient, and to also assist in identifying their parking location.

From initial meetings and thorough discussions of all possible implementations, we decided that the most effective and environmentally friendly approach would be the following structure:

- **Old mobile devices which would act as sensors for the parking system**

These would be positioned above parking spots in the car park, and would be connected to a constantly on internet and power supply

- **An online based database**
  This would be a virtual storage of the parking locations as well as the status of the parking spots, either being occupied or empty.
- **A mobile application**
  This would be used as a point of interaction with the parking system. It will show the statuses of the parking spots, as well as provide querying interactions, such as finding a parking spot that is at a convenient location and also saving the spot that they've parked at.

Based on the experiences and capabilities of the team, we chose to develop for the Android platform, along with a server-side script implementation to handle the map and pushing of real time map data to user's phones. For the sensor, a mobile app would be built in Java that would use image processing techniques which would detect the presence of a car.

Following the decision of the platform and technologies to be used for this project the proceeding sections of the report details out what was achieved and how this was achieved.

2 DEVELOPMENT METHODOLOGY

The proposed project that we are going to undertake is a system designed to improve a targeted community, Nottingham Malaysia Campus and overcome certain issues like the usability of the parking system and the overwhelming electronic waste. We decided to adopt the agile methodology as our software development methodology because it is best suited to our scenario. The main reason we chose agile over a conventional process model is that we will deliver better in a working environment which is centered on seamless communication and interaction between team members since we are only a four-man team. With such a small organization, it is always handy to arrange informal meetings to discuss on our progress and face-to-face communication can yield comparatively higher efficiency in our productivity as the team can remain compact and closely-knitted all the time. Agile methodology encourages customer involvement in the software development process. Hence, with the customers as a part of the team, they as the domain expert can help in our requirement analysis significantly and constantly reviewing our progress to keep us on track. We need a process model which is flexible and can easily accommodate changing requirements. Since there are some uncertainty in the requirement and design, with agile methodology, we can better adapt to uncertainty faced during requirement analysis. Since we can

iteratively repeat our entire development cycle, we can review our system requirement frequently and make changes if there is a need.

We plan to practice the five values of Extreme Programming on top of Agile Methodology in our system design and implementation. The five values are communication, simplicity, feedback, respect and courage. These values are founded upon the basis of agile methodology. We will prioritize communication with our stakeholders and get valuable feedback from them by delivering frequent small releases. We will also encourage simplicity in design and coding so that refactoring of code will be very much easier whenever there are changes in the requirements. Fundamentally, we respect each other and appreciate the effort by each and every member of the team.

Extreme programming is an iterative development process. In order to facilitate our progress, we will have an informal and a formal meeting per week. A formal meeting is held with our supervisor and stakeholders while an informal meeting is just held among the group members. In each informal meeting we will review on previously assigned tasks and what needs to be done next. We will make sure that each group member has been assigned task at the end of every informal meeting. The purpose of a formal meeting is to report our progress to our supervisor and also seek guidance from him.

2.1 PRACTICES

2.1.1 ITERATIONS

Unlike conventional waterfall development methodology, our development phases are far more flexible as we continuously iterate between planning and implementation phases. Being more specific, we will go through requirement analysis, software design, coding and testing each iteration. However, the structure of these phases does not mean that we have to stay rigid and follow the sequence of the phases. We welcome active feedback from our stakeholders and we will not be afraid to implement further changes to our system requirements. Having said so, we therefore are not restricted from jumping back to previous stages if we feel like making some adjustment to it. Minutes of meeting are recorded and documented after every meeting to keep track of progresses and status.

2.1.2 Communications

Our group members maintain close communication with each other either through face-to-face or online conversation. This is essential so that each member is easily approachable and there is transparency in everyone's working progress. Off-school, we make good use of the popular smartphone instant messaging service, Whatsapp to conduct discussion regarding project tasks and also meeting schedule. In this Whatsapp group chat, everyone has an equal opportunity to voice out their personal opinions and help each other out. For task delegation, we use a Web-based application called Asana to assign tasks to every group member immediately after each informal meeting. This provides a guideline and reminders for us to work on our parts as the complexity level rises, so that there will be no excuses for us to not complete our tasks in each sprint cycle.

2.1.3 SIMPLICITY

We exercise simplicity in design, from source code to user interface. Simplicity is an important aspect to consider in our software design because as the system becomes too complex, the dependencies within the system may cease to clear without proper designing. With Java as our primary language, we can make the best out of the object-oriented paradigm by maximizing code reuse and modularity. We will also employ the functionality of the existing libraries to avoid creating something redundant which is already out there available for us.

2.1.4 PAIR PROGRAMMING

Pair programming is definitely the essence of extreme programming methodology. Since solo programming can be very much error prone and less efficient, pair programming will minimize these weaknesses by having an extra brain and pair of eyes doing the same job. We will divide our group into two pairs and we will run our coding tasks in parallel on separate program components or modules. For each pair programmers, one will be doing the coding while another will overlook the entire code blocks and scan for errors and bugs to be fixed. By doing this, not only we will speed up the pace tremendously, the practice will also yield higher quality code.

3 SYSTEM SPECIFICATION

3.1 UPDATED SYSTEM SPECIFICATION

3.1.1 FUNCTIONAL REQUIREMENT

1. The sensor application should be able to detect if there is a vehicle parked on its assigned parking bay.
   1.1. The sensor application will make use of the phone camera to take picture once per every 7 seconds interval.
   1.2. The sensor application will compare the latest two consecutive images for the difference in pixel values. If the difference exceeds a pre-set threshold, the application will decide that vehicle is detected.
   1.3. The sensor application will update the online database status if a vehicle is detected.

2. The mobile application requires Internet connection to operate.
   2.1. The mobile application will automatically establish Internet connection when started.
   2.2. The mobile application will not crash if it fails to connect to the Internet.
      2.2.1. The mobile application will display a dialog box to prompt the user to check for network connection.
      2.2.2. The mobile application will not be able to display and run all Internet bound activities.

3. The mobile application should be able to display a small-scale prototype car park map.
   3.1. The map layout should be displayed automatically when users start the application.
   3.2. The map layout should represent red and yellow parking zones correctly.
      3.2.1. Red parking bays should be labelled with red colour.
      3.2.2. Yellow parking bays should be labelled with yellow colour.
   3.3. The map layout should indicate all map objects with name labels.
      3.3.1. Parking bay objects should be labelled with parking bay number.
      3.3.2. Building objects should be labelled with the building name.
   3.4. The map layout should indicate which parking bays have been occupied.
      3.4.1. Occupied parking bays should be greyed out.

4. The mobile application should be able to present a real-time status of the car park information.
    4.1. The mobile application shall constantly pull data from the server to ensure it always has the latest instance of the car park information.
    4.2. Parking bay will be shaded once it is occupied.
    4.3. Parking bay will be unshaded once it is available again.

5. The mobile application should allow the users to interact with the user interface.
    5.1. The mobile application should enable users to navigate between Activities.
        5.1.1. A navigation drawer will be expanded by the user swiping from the left edge of the screen or by touching the app icon on the action bar.
        5.1.2. The navigation drawer will have the options of pages that the users can navigate to.
        5.1.3. The navigation drawer shall be able to be dismissed.
    5.2. The map view should be zoomed in when users pinch open.
    5.3. The map view should be zoomed out when users pinch close.
    5.4. The map view should be scrollable with swipe or drag gestures.

6. The mobile application should allow the users to find empty parking spots according to their preferences.
    6.1. The mobile application should allow the users to input query.
    6.2. The query details shall encompass parking type (red / yellow) and a designated building.
    6.3. The mobile application will search for any empty parking bays within closest proximity to the selected building and highlight the results in the map.

7. The mobile application should enable the users to save their parking spot locations.
    7.1. The mobile application will highlight the parking bay saved by the user to indicate its location on the map.
    7.2. The saved preference shall persist even after the application was terminated, and always available to be shown to the users whenever they need it.
    7.3. The mobile application should allow the users to overwrite the previously saved parking location with a new parking location.

8. The mobile application should provide user guide to assist first-time users.

8.1. User guide will be demonstrated in the form of a series of popup messages.
8.2. User guide will give the users an overview of the application's features and step the users through their first user experience.

9. The mobile application should provide options for the users to modify its settings.
   9.1. The mobile application should allow the users to turn on or turn off the user guide feature.
   9.2. The mobile application should allow the users to clear the saved parking location data.
   9.3. The mobile application should allow the users to view the open source licenses of the external libraries that are adopted during the project development.

3.1.2 NON-FUNCTIONAL REQUIREMENT

1. Usability
   1.1. The sensor system will only be operating during the office hours, which is from 8 a.m. to 5 p.m., as the parking issue will not remain a problem off office hours.
   1.2. The mobile application should be usable by people of all ages, gender, races and cultures and also education background especially to those who are in need of parking service in campus.
   1.3. The mobile application should have a simple and intuitive user interface so that even new users can easily learn how to use it in a few runs.
   1.4. The users require Internet access on their mobile phones in order to activate the core functionality of the mobile application.

2. Performance
   2.1. The system need to have an algorithm to detect vehicle and it need to be optimized enough to be able to run on low-end android phones.
   2.2. The sensor algorithm must be smart enough to not falsely analyse data if encountered with mild disturbance from the environment.
   2.3. The maximum response time from the system to the users shall not be more than 10 seconds as the sensor keeps replying every 7 seconds while the user app polls the server every 1 second.

2.4. The sensors need to communicate with the server via wireless network. Therefore, parking lots will need to be equipped with Wi-Fi coverage with internet connection.
2.5. The mobile applications need to be coded and supported in Java as it is the targeted development environment in Android. The database backend will be running on PHP-MySQL platform.
2.6. The server shall be able to handle the network traffic and data operations from a saturated car park situation and also a large user base.
2.7. The system shall be capable of handling the actual implementation of a full-fledged university car park capacity.

3. Dependability
   3.1. The Android phone shall be able to carry out its required functions properly, such as capturing and processing images.
   3.2. The Android phones (sensors) shall have sufficient power supply to sustain the operations throughout their working hours.
   3.3. The shelter for the sensor shall be strong enough to shield the device from external damaging factors.
   3.4. The user application shall always be in sync with the server.
   3.5. The features of the user application shall not provide misleading information to the users.

4. Security
   4.1. The server shall be secured so that the information kept and transmitted across components will not be intercepted and sabotaged.
   4.2. The server shall only provide access to the system administrators who can provide the private key to log in using SSH.
5. Compatibility
   5.1. The user application is only available on Android platform.
   5.2. The user application supports a minimum API level of 8, which is Android 2.2 – 2.2.3 Froyo.
   5.3. The user application supports up to the latest API level 19, which is Android 4.4 KitKat.

6. Portability
   6.1. The user application shall be able to be installed in all Android devices from Android version Froyo to KitKat.
   6.2. The user application shall be able to function as targeted as long as the device is connected to the Internet.

3.2 UPDATED DESIGN SPECIFICATION

3.2.1 USER INTERFACE

The final interface that was implemented had slight modifications from the initial prototypes that were developed, in various ways:

1. It was discovered that the method for navigation between the app's pages were conflicting with the design guidelines suggested by Google, so the main navigation method decided on was through the use of navigation drawers.
2. A proto-type map was implemented, instead of the entire university parking.

USER INTERFACE AT LAUNCH

At launch, the application checks if the device is connected to the internet.

In the event that it isn't, the following prompt is displayed:

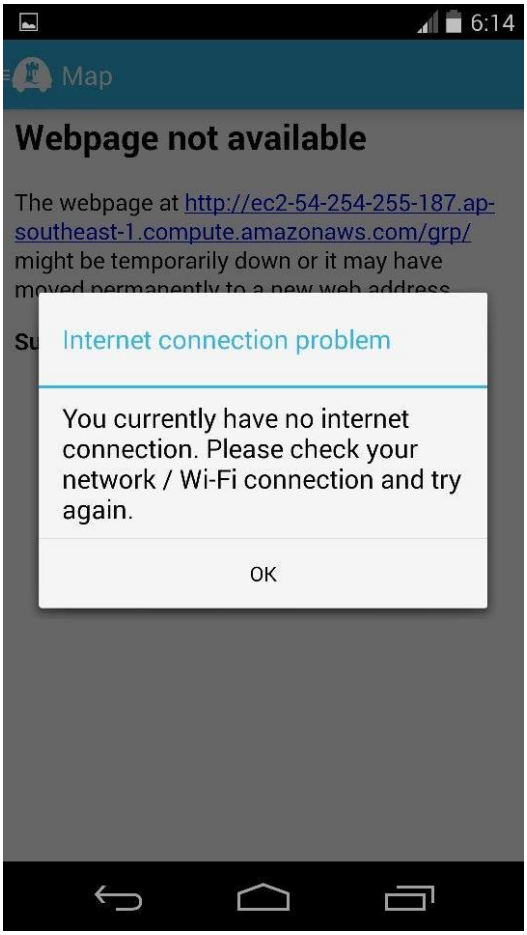


Figure 3.1: User Interface 1

After dismissing the notification, the launch of the app is continued although the pages requiring an internet connection will be non-functional. It has been implemented in this way, as crucial information regarding the parking spot is saved locally.

NAVIGATION DRAWER

At the left of the top action bar, a toggle is provided, which on touch opens the navigation drawer as illustrated below:

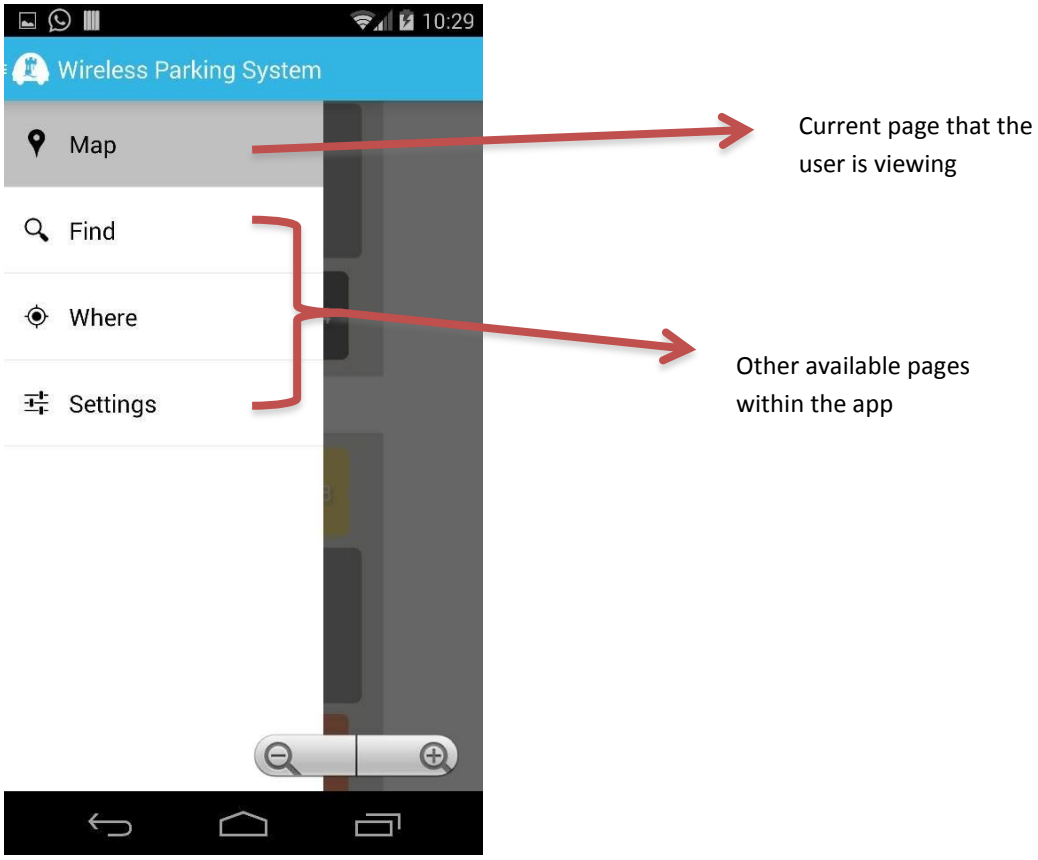


Figure 3.2: User Interface 2

MAP PAGE

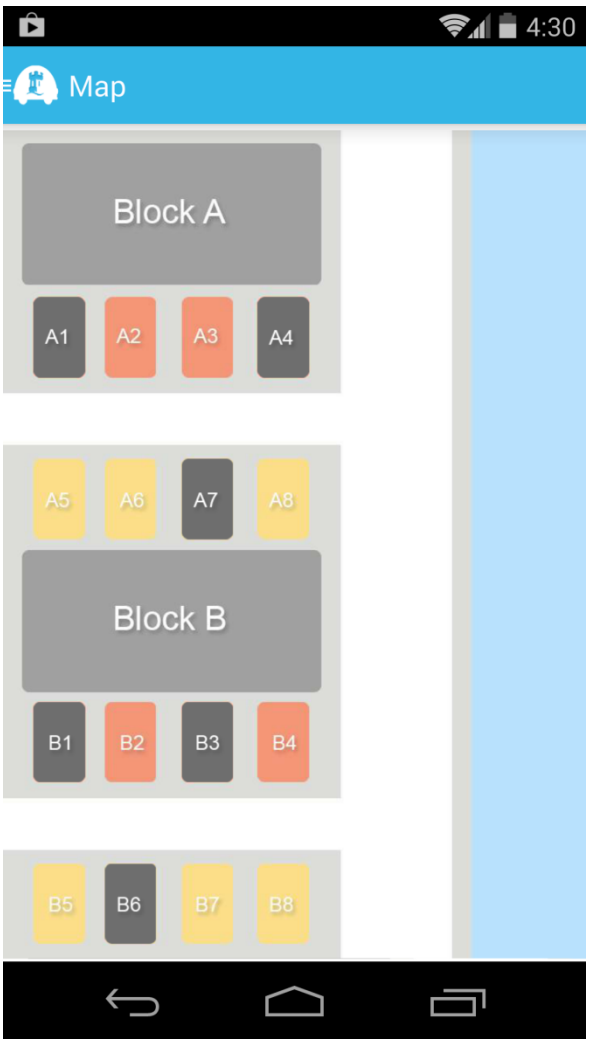


Figure 3.2: Map User Interface

At launch, the first screen that the user sees is the map of the parking lot, along with their availability. A key is used to help for the first time users, with the following configuration:

- Red – The available parking spots usually reserved for the staff and lecturers on campus.
- Yellow – The available parking spots for both staff, lecturers and students
- Black – An occupied parking spot.

FIND PAGE

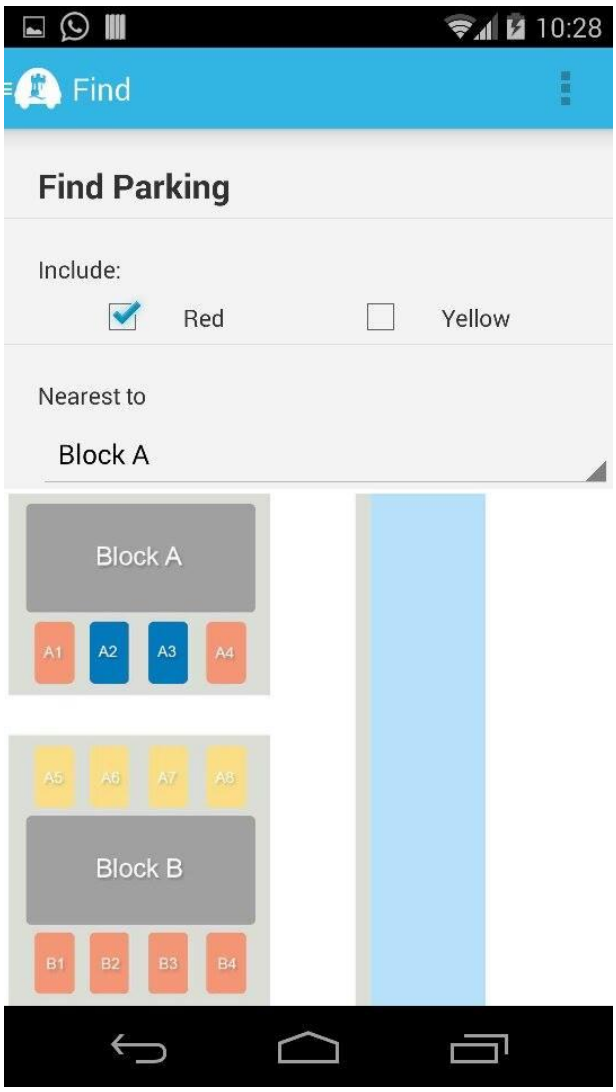


Figure 3.4: Find User Interface

The find page within the app is used to search for a convenient parking location for the user, depending on 2 criteria:

- The type of parking
- Proximity to building that they would like to go to

The results are then highlighted in a dark blue colour, which the user may go to.

WHERE PAGE

On this page, details of parking location that the user has parked in are saved. Once the user has parked, he selects the parking spot to save his location at. The parking is spot is then saved, listed and highlighted in a dark blue colour.

An example is as shown below:

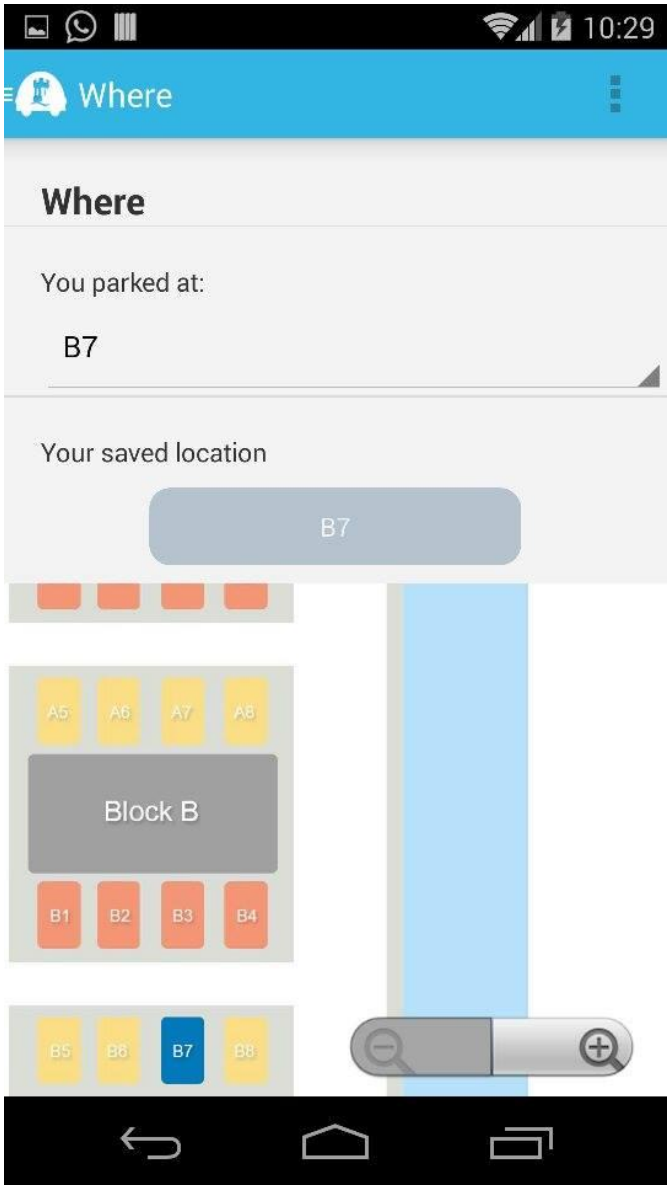


Figure 3.5: Where User Interface

The parking location is saved, even if the user exits the application.

SETTINGS PAGE

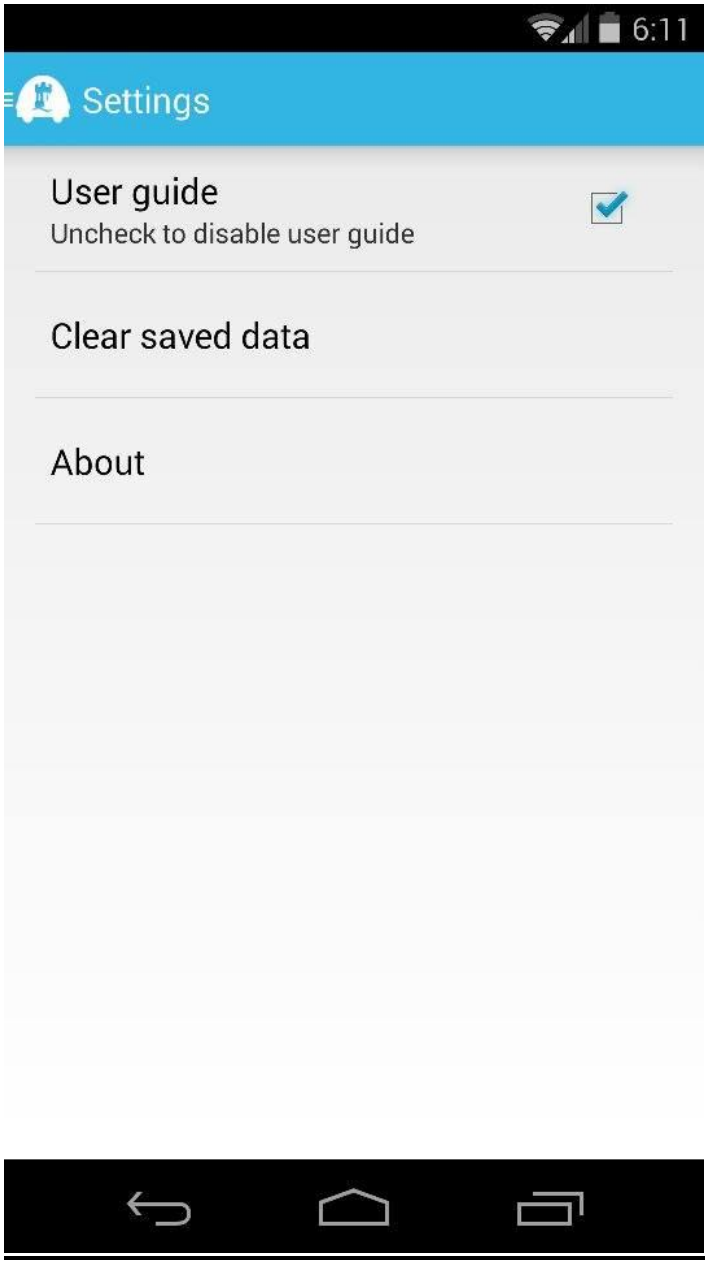
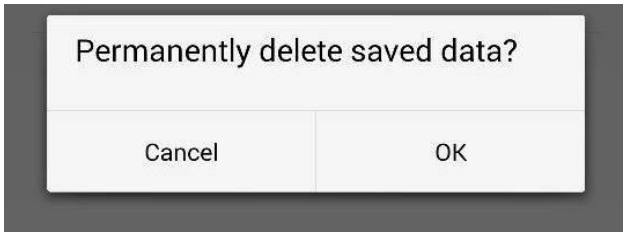


Figure 3.6: Setting User Interface

This is the settings page for the mobile app. It contains:

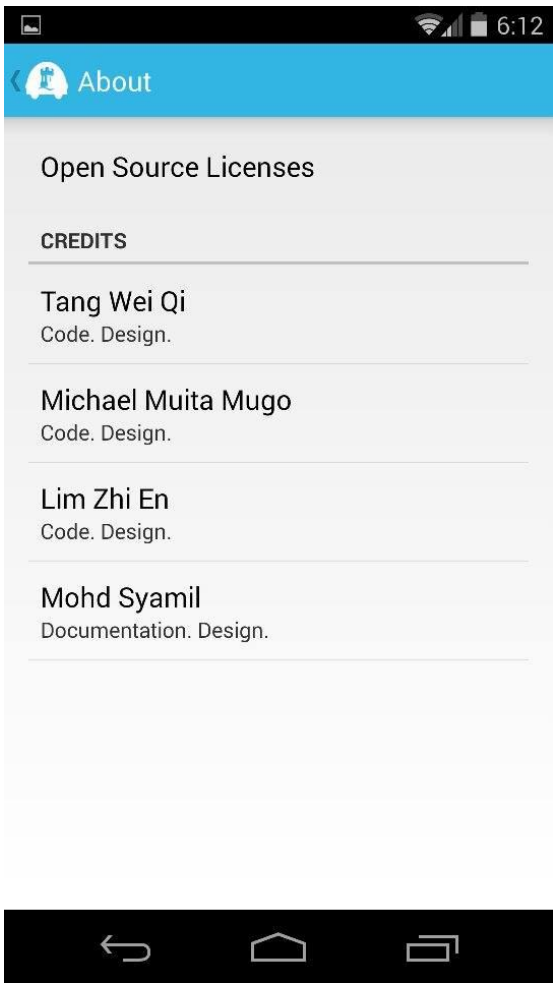
- A toggle for the user guide, which is usually visible during each start-up as notifications. It is implemented as shown below:

- A button to clear the locally saved information about the parking spot of the user. It is accompanied with the following prompt:



- And a button linking to an about page.

ABOUT PAGE



This is the About page, it details out:

- The credits for the mobile application
- The open source code implemented by the application.

Additionally, it has been set out to intentionally go back to the Settings page, instead of the drawer.

Upon selecting the Open Source Licenses button, the following notification is displayed:

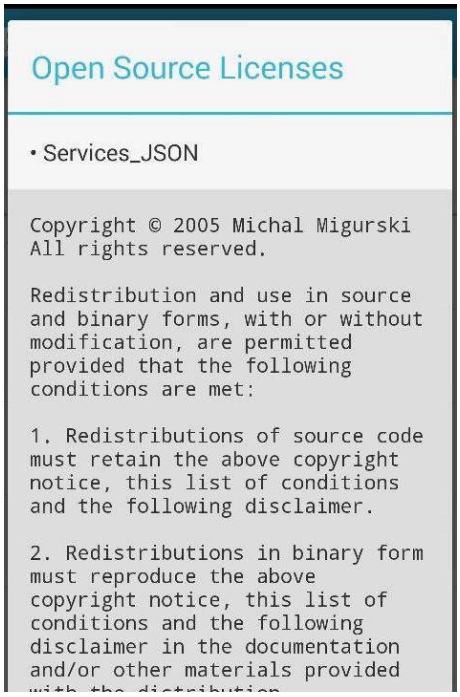


Figure 3.7: About User Interface

4 SYSTEM IMPLEMENTATION

4.1 SENSOR

With more android phones sold worldwide, there is an increasing abundance in used android phones which are not reused. Out of them many ways to determine a car park status, our team decided to utilize the camera module in these recycled phones to determine the availability of a car parking bays. Basically the recycled android phone itself will act as a sensor which is connected to the Wireless Parking System database through the internet in any means of connectivity. The detection of a car on a parking using a camera module requires an image processing algorithm that is practical enough to runs on used android phones where the hardware capabilities are often outdated compare to the latest technologies available in the market. This poses a challenge to our team to research on and seek on the best way to apply car detection using an android phone.

From our past experiences in the Introduction to Image Processing module, there are numerous ways in determining an existence of an object which is a car in our case. A few of them include edge detections and others advanced algorithms. Since hardware capabilities are limited in our case, we decided to go with the simplest algorithm in detection where we determine the differences between consecutive images and figures how much changes are there. The sensor application will first convert two images being compared into greyscale and compares every pixel with its relative pixel on the other image and determine the differences in greyscale intensity. If the absolute of the differences between those pixels are higher than a pre-set threshold value, the pixel will be labelled as a changed pixel. The process repeats itself with the rest of the pixels on the images and a changed pixel percentage will be calculated using the formula:

$$\frac{total\ no.\ of\ pixel\ labeled\ changed}{total\ no.of\ pixel} \times 100\%$$

The result represents the percentage of pixel changed according to the intensity threshold predetermined. Higher percentage indicates that there is a huge change in those pictures compared. As the position of the sensors is fixed in a way that the camera monitors a particular parking bay, there won't be any big changes in the consecutives images unless there is a car introduced into the camera frame. With this theory, the changes in status of a parking bay can be determined by a huge percentage of pixels changed.

By the implementing the algorithm mentioned above with a flipping switch, the status of the parking bay could be determined in the condition that the parking bay is vacant in status during initialisation of the sensor and the first

significant change in pixel indicates there is an introduction of a car into the bay. For example, sensor at a particular parking bay is initialised and every frequent comparison of the relative images indicates minimum changes will show that that particular bay is not yet occupied by any car and vice versa. Once the status of the bay is taken, the next huge change would indicate the car has left and yet again it is vacant.

Figure 4.1: Sensor User Interface

The image above shows a screenshot of an operation in progress sensor. On the top left corner, there is a spinner that allows the administrators to select the parking bay id that the particular sensor checking. On the top right corner, there is a flash toggle button that toggle flash light on and off for various situation. There log running explains the process of the sensor application and its status.

4.2 USER APPLICATION

4.2.1 USER INTERFACE

The application follows the fragmentation guidelines of Android. It contains 4 fragments, along with an adapter and model to manage the navigation drawer and a main activity to connect all the components together. Along with this, various styling information and resources were modified to implement the application in a single frame layout, to enable navigation through a drawer.

BACKGROUND

Following the upgrade in the Android ecosystem into the Jelly bean and future versions, Google implemented a new standard in navigation for all mobile applications. It entails the representation of the portions of a user interface in an activity. These representations are known as fragments.

The following diagram illustrates the concept behind the use of fragments for navigation:

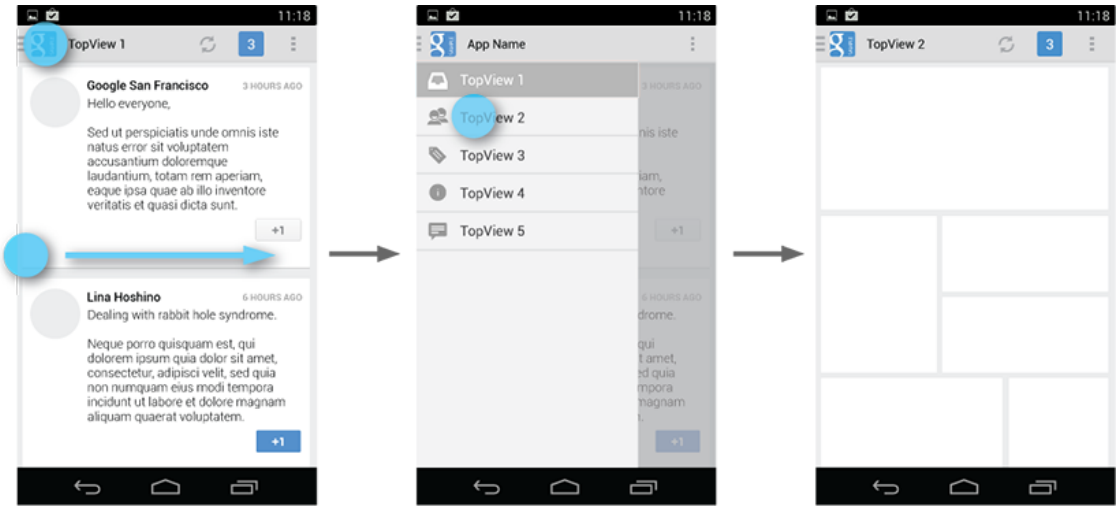


Figure 4.2: Drawer

The user accesses the drawer by swiping from the left edge of the screen or by touching the application icon on the action bar. As it opens, it overlays the content but not the action bar.

The panel that displays app's main navigation options on the left of the screen is implemented by declaring the main user interface component as a DrawerLayout object as the root. Inside it, two child views are used:

- FrameLayout to contain the main content (This is populated by the fragments at runtime)
- ListView to contain a list of the various pages available within the app

At start-up, the navigation drawer's list of items are populated, by the use of an adapter which basically provides a series of calls that connects all the various resources, such as strings, styling and images (stored in various sections of the app) together. The items are then associated with an action listener, which connects to changes the fragment viewed on the main activity. They are pre-assigned by default, and upon selection, the main activity's fragments are changed.

LAYOUTS OF INDIVIDUAL PAGES

Each Fragment has been assigned a unique xml layout, often containing multiple nested linear layouts, weights and alignments to ensure compatibility with various screen sizes.

4.2.2 MAP RENDERING

How do we render the car park map on our Android user app?
We figured a brilliant way which simplifies everything, which is to use the WebView and web scripting for the map design. The graphical layout of the map is designed using Photoshop. On top of that, we created a library of sprites as masks for each individual parking bay. Each sprite will have the exact same coordinates as the corresponding parking bay in the original map's blueprint. The idea here is to overlay the entire collection of sprites on top of the original blueprint and manipulate the visibility of each mask by using JavaScript, in accordance to the current instance of the parking space availability. The content of the map is defined as <img> elements in HTML. A bit of CSS is used in setting the position, layering and display options of the sprite images. Using the class attribute, all sprites are classified as one category to facilitate manipulation later. Each sprite will be assigned an id as of the Sensor ID in the database. Hence, each sprite is mapped with its corresponding database entry.

How do we make the map instance real-time?
This is where AJAX comes into place. AJAX, or in full Asynchronous JavaScript, is the art of exchanging data with a server, and updating parts of a web page, without reloading the whole page.

How it works?

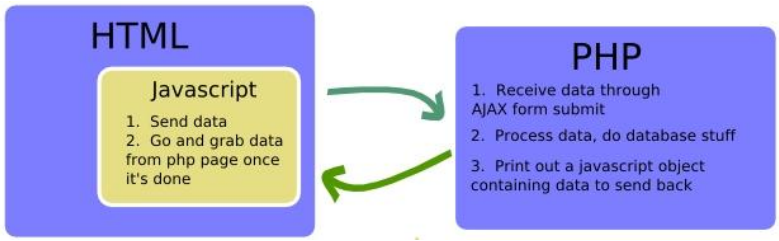


Figure 4.3: Simple AJAX

JavaScript submits a form to a PHP page using AJAX. As soon as the server-side PHP has received the data through AJAX form submit, it will process them and carry out database operations. After PHP is done with works, it will print out a JavaScript object containing data to send back to the client-side JavaScript. The JSON-PHP module can convert a PHP array of key-value pairs to this JSON object. JSON is a lightweight data-interchange format based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. It is easy for humans to understand and for machines to parse. JSON is a text format that is completely language independent, thus makes it an ideal data-interchange language.

Back to our case, we have two JavaScript functions, ajax_request and ajax_response. Ajax_request will submit the form to ajax_request.php through AJAX and then calls the ajax_response function when AJAX data has been retrieved. The retrieved data will be passed on to ajax_response as a JSON object parameter. The JSON object returned contains all key-value pairs from the database with key being the Sensor ID and value being the Status at that particular instance. Within the ajax_response function, we get hold of all the sprites by class name, perform cross-reference checks with the retrieved data and then toggle the visibility of each sprite according to its current status. In order to have the map instance real-time, we used JavaScript's setInterval method to repeatedly poll ajax_request.php for the latest changes in the database and then update the web page accordingly.

4.2.3 FIND FEATURE

Search algorithm

The logic behind the Find algorithm is by defining distance metadata. We first calculated the geographical distance between parking bays and buildings and produced a scaled measurement of the distance. The scaled values were added to the database as distance metadata. The Find algorithm will search

for empty parking bays with priority given to those nearest to the queried building based on the distance metadata.

Parameters passing
The essence of the Find feature is to find solution based on custom user preferences. In order to accomplish that, we need to send the user queries across to the server so that the server-side processing can be executed accordingly. We figured to use the HTTP GET method. Within each Find session, the user option variables are generated into a GET query string to be sent over to the find.php on the server. Upon receiving the GET parameters, find.php will utilize them to carry out the Find operations and return the search results to the clients.

4.2.4 WHERE FEATURE

Shared Preferences
The Where feature was designed to let the users save their parking locations locally in their devices. Using the Android SharedPreferences API, we were able to program it to save the user data in the memory stack as key-value pairs. The saved data will be kept in memory even after the application process was killed. To our benefit, Shared Preferences allows the saved key-value pairs to be overwritten with new data. As a result, our application users can always renew their parking locations by simply reselecting the parking bay from the Spinner[1].

4.2.5 SETTINGS CONFIGURATION

The Settings fragment was an extension of the PreferenceFragment. It is structured as a hierarchy of Preference objects via XML. Visually, it shows a list of preferences of which configurable by the users. These preferences will automatically save to SharedPreferences as the user interacts with them. Our application settings provide the following preferences:

User guide
This is a CheckBoxPreference. The user guide feature is targeted towards the new users of our application. The user guide is demonstrated by a series of Toast[2] messages displayed on the user interface. It provides step-by-step instructions to assist the new users in using the features of our application.

[1] Dropdown menu in Android's term

[2] Popup message in Android's term

Users have the freedom to enable or disable this feature by checking or unchecking the checkbox.

Clear saved data
This is a DialogPreference. Upon selecting this preference, users have the choice to remove the saved parking location from the Shared Preferences memory stack.

About
This is a PreferenceScreen, which brings to a subscreen. The subscreen consists of an Open Source Licenses DialogPreference and a Credits PreferenceCategory.

4.3 CONNECTING ANDROID TO MYSQL

In order to allow our sensor app to directly update the centralized database in real-time basis, we need to devise a mean of communication between Android and MySQL. Through research we found a way to do so, which is to use PHP as the intermediate platform to connect Android with MySQL.

Basically, this is how the above architecture works. The Android app calls a PHP script to perform a data operation, such as “Update”. The PHP script then connects to your MySQL database to perform the operation. So the data flows from the Android app to the PHP script then finally is stored in the MySQL database.

4.4 ONLINE SERVER

The nature of our project requires a centralized database for simultaneous data retrieval from multiple user application. This criterion has set up our requirement to acquire an online server where the MySQL database will be populated. Aside setting up an online server, we can hardly find any other easier ways to achieve this goal. While keeping in mind that the online servers usually do not come without cost, we started our “hunt” for an online server which will not cost us a single penny at all. Fortunately, we came across Amazon Web Services (AWS), a collection of remote computing services that together make up a cloud computing platform, offered over the Internet by Amazon.com. One of the major services provided by AWS, which is the Amazon Elastic Compute Cloud (EC2), perfectly fulfils our needs.

4.4.1 AMAZON ELASTIC COMPUTE CLOUD (EC2)

EC2 provides resizable computing capacity in the AWS cloud and allows us to launch as many or as few virtual servers, known as instances, as we need.

Instances and AMIs
An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, you launch an instance, which is a copy of the AMI running as a virtual server in the cloud.

Regions and Availability Zones
Amazon EC2 is hosted in multiple locations world-wide. These locations are composed of regions and Availability Zones. Amazon EC2 allows you to place resources, such as instances, and data in multiple locations. In our case, we launched our instance in Asia Pacific (Singapore) Region (code: ap-southeast-1) as it is geographically the nearest location to our working station.

Amazon EC2 Key Pairs
Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. The public and private keys are known as a *key pair*.
To log in to our instance, we have created a key pair (myserver.pem). We have to specify the name of the key pair when we launch the instance, and provide the private key when we connect to the instance. Linux/Unix instances have no password, and we use a key pair to log in using SSH.

Security Groups
A *security group* acts as a virtual firewall that controls the traffic for one or more instances. We have selected Bitnami LAMP stack as the security group of our instance because it is completely integrated and configured. It includes Apache, MySQL, PHP and phpMyAdmin and all of the other software required to run each of those components. As such, Bitnami LAMP stack greatly simplifies the development and deployment of our PHP.

AWS Free Usage Tier
The interesting point is, AWS is offering free usage tier to help new customers get started in the Cloud. The free tier can be used for anything you want to run in the Cloud, including EC2 and it will be available for 12 months following our AWS sign-up date. This is the deciding factor that makes us choose Amazon EC2 as our online server because it will be free-of-charge for one year, at least covering our requirements throughout the entire project timeframe.

(AWS)

CONNECTING TO SERVER

Linux/Mac users

As Linux and Mac OS X come bundled with SSH clients by default, in order to connect to the server using SSH, the Mac users in our group need to open the terminal and type the following:

ssh -i myserver.pem bitnami@ec2-54-254-255-187.ap-southeast-1.compute.amazonaws.com

Windows users

Windows does not come with a bundled SSH client by default. So, those of us who are Windows users will have to use the freely available Putty utility to connect to the server. SSH key in PPK format is also needed for authentication.

Description of	EC2 Instance
Public DNS	ec2-54-254-255-187.ap-southeast-1.compute.amazonaws.com
Public IP	54.254.255.187
Availability zone	ap-southeast-1a
Security groups	LAMP
Key pair name	myserver

Table 4.1: Description of EC2 Instance

4.4.2 MYSQL DATABASE

We need a database to store all the data required to sustain the operation of our system. EC2 has provided us a very easy-to-use phpMyAdmin infrastructure to handle the administration of MySQL over the web. The database is named “wireless_parking” and it is set up in the localhost of our EC2 instance. It contains only one table, the “main_parking”. We will give you a visual representation of our database in a table form as shown below.

Sensor_ID	Zone	Type	Status	Distance_A	Distance_B
A1	A	Red	1	1	2
A2	A	Red	0	1	2
A3	A	Red	0	1	2
A4	A	Red	1	1	2
A5	A	Yellow	0	2	1
A6	A	Yellow	0	2	1
A7	A	Yellow	1	2	1
A8	A	Yellow	1	2	1
B1	B	Red	0	3	1
B2	B	Red	0	3	1
B3	B	Red	1	3	1
B4	B	Red	0	3	1
B5	B	Yellow	0	4	2
B6	B	Yellow	1	4	2
B7	B	Yellow	0	4	2
B8	B	Yellow	0	4	2

Table 4.2: Database Table

Field Description	
Sensor_ID	Primary key for each record. Equivalence of parking bay ID. Each parking bay corresponds to a unique Sensor_ID belonging to each sensor device.
Zone	Car park zone. Implied by the prefix of the Sensor_ID.
Type	Category of parking lot. Red signifies Student Parking. Yellow signifies Staff Parking.
Status	Availability of each parking bay. 1 signifies Occupied. 0 signifies Vacant.
Distance_A	Metadata of distance from Block A. Scale of 1 to 4 with 1 signifying the nearest and 4 signifying the furthest.
Distance_B	Metadata of distance from Block B. Scale of 1 to 4 with 1 signifying the nearest and 4 signifying the furthest.

Table 4.3: Field Description

5 OVERVIEW OF DEVELOPED SOURCE CODE HIERARCHY

5.1 DIRECTORY STRUCTURE OF AN ANDROID PROJECT

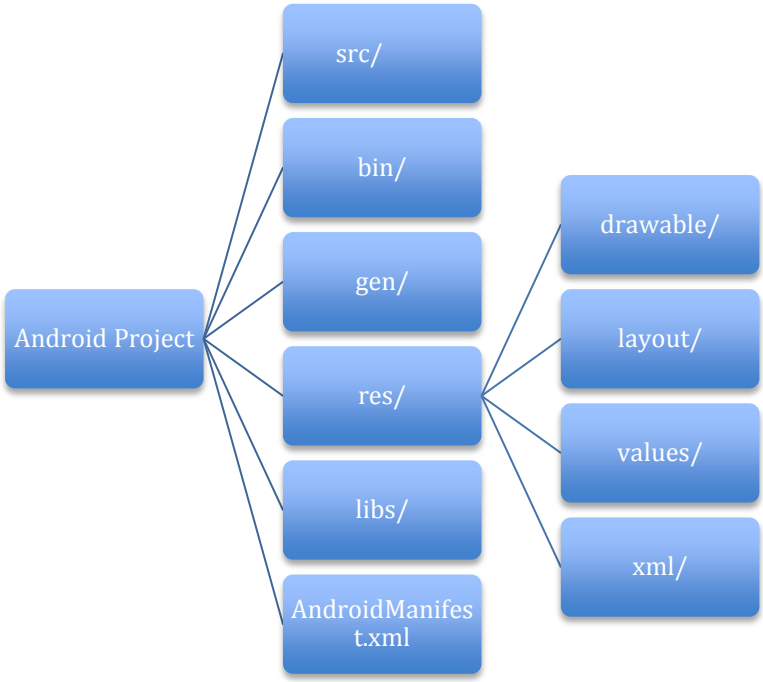


Figure 5.1: Android Project

Android Projects
An Android project is the container for your application's source code, resource files, and files such as the Ant build and Android Manifest file. An application project is the main type of project and the contents are eventually built into an .apk file that you install on a device.

src/
Directory for your app's main source files. By default, it includes an Activity class that runs when your app is launched using the app icon.

bin/
Output directory of the build. This is where you can find the final .apk file and other compiled resources.

gen/
Contains the Java files generated by ADT, such as your R.java file and interfaces created from AIDL files.

res/
Contains several sub-directories for app resources. Here are just a few:

drawable/
Directory for drawable objects (such as bitmaps)

layout/
Directory for XML files that are compiled into screen layouts (or part of a screen)

values/
Directory for XML files that contain a collection of resources, such as string and color definitions.

xml/
For miscellaneous XML files that configure application components.

libs/
Contains private libraries.

AndroidManifest.xml
Every application must have a manifest file in its root directory. It presents essential information about your app to the Android system, information the system must have before it can run any of the app's code.

5.2 PROJECT DIRECTORY OF SENSOR MODULE

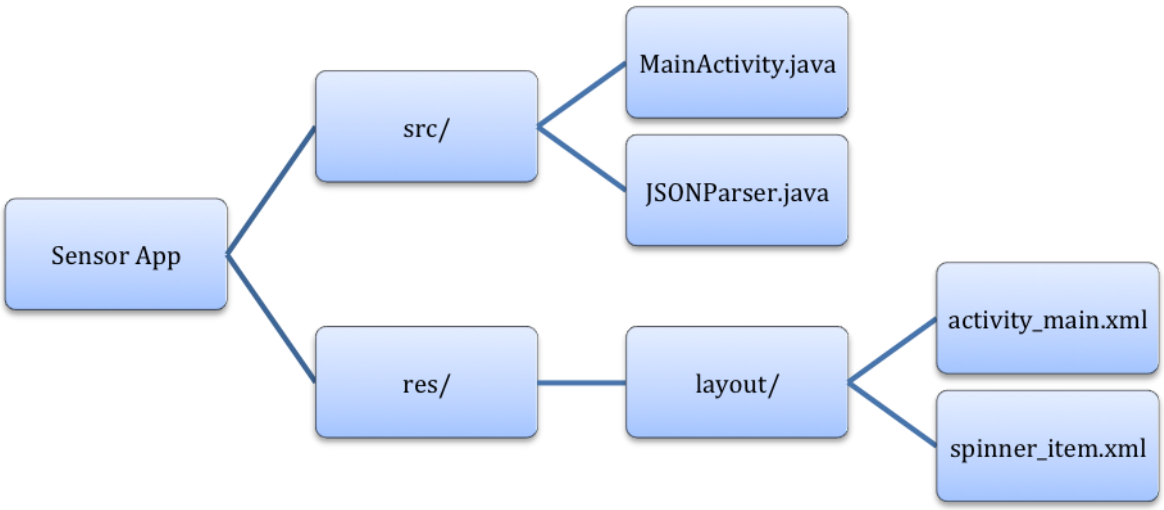


Figure 5.2: Sensor App Directory

Android Web Service
In the sensor app, we created an Android Web Service to transmit data over the Internet to the server-side PHP scripts via HTTP POST request. The

response hence replied by the server will be encoded in JSON data structure because it is very light weight, structured, easy to parse and much human readable. These characteristics make JSON the best alternative to XML when your Android app needs to interchange data with your server.
In order to prevent the HTTP requests from hogging up the UI thread, we are getting the JSON by making the HTTP calls in AsyncTask. With AsyncTask, the HTTP calls are executed in the background thread while the results are then pushed to the UI thread to be displayed.

JSON Parser

Since we are getting JSON as server response, we need a JSON parser in our Android sensor app to correctly display the response from the HTTP calls made. For the JSON parser, we have adopted an open-source JSON Parser class source code from AndroidHive.
Within the doInBackground method of the AsyncTask, two parameters are encapsulated into NameValuePairs to be sent over through HTTP POST request, namely the Sensor ID and the Status. Status parameter will be a boolean value indicating the vacancy of the specified parking bay represented by the unique Sensor ID parameter. These NameValuePairs along with the destinated URL and the HTTP method will be passed to the JSON Parser. JSON Parser will send the HTTP request to the specified URL and return a JSONObject of the server response. Hence, in the onPostExecute method of the AsyncTask, the sensor will output a server response string to inform the users of the success of the data transmission.

5.3 PROJECT DIRECTORY OF USER APPLICATION

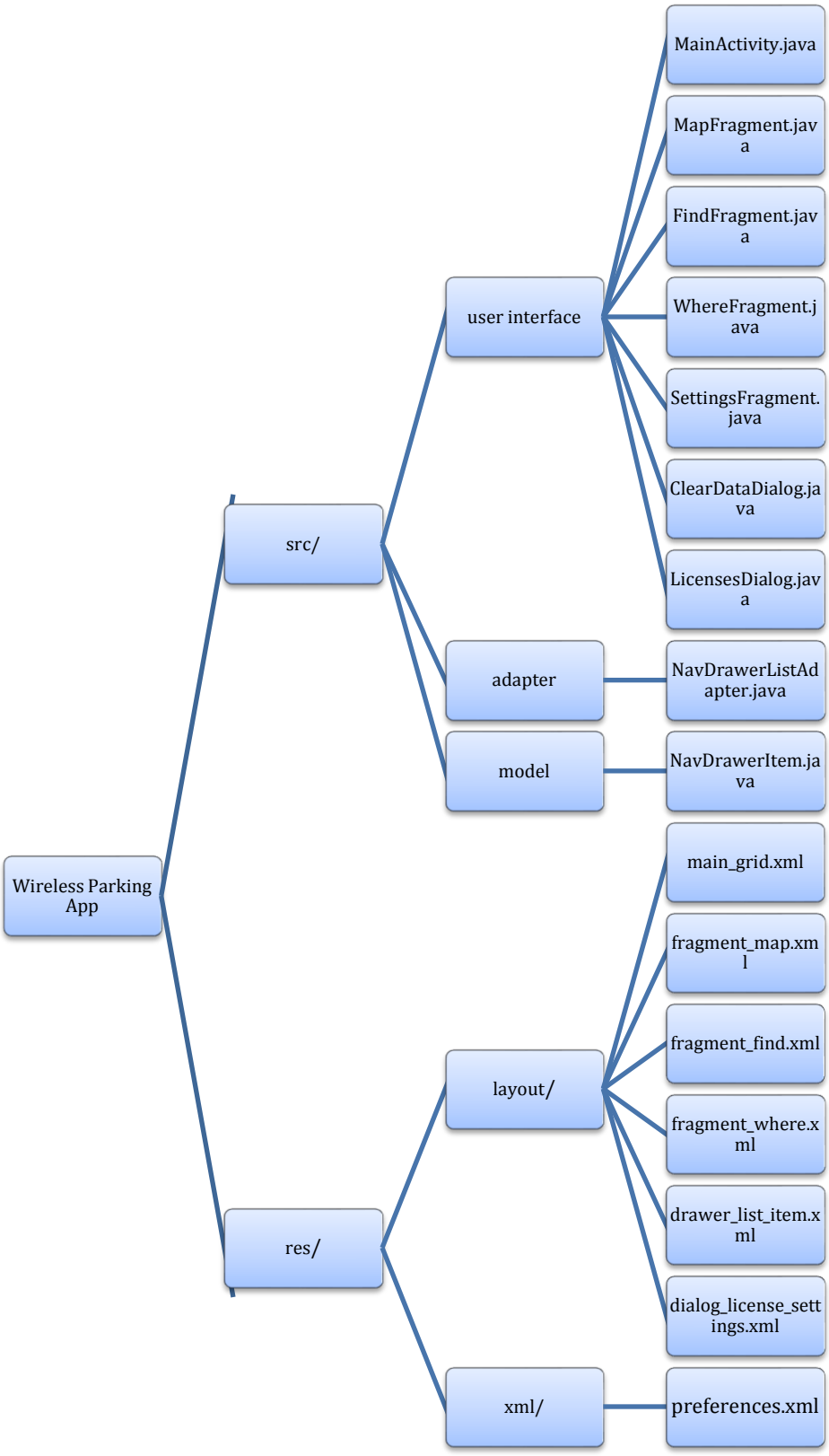


Figure 5.3: Wireless Parking App Directory

5.4 SERVER DIRECTORY

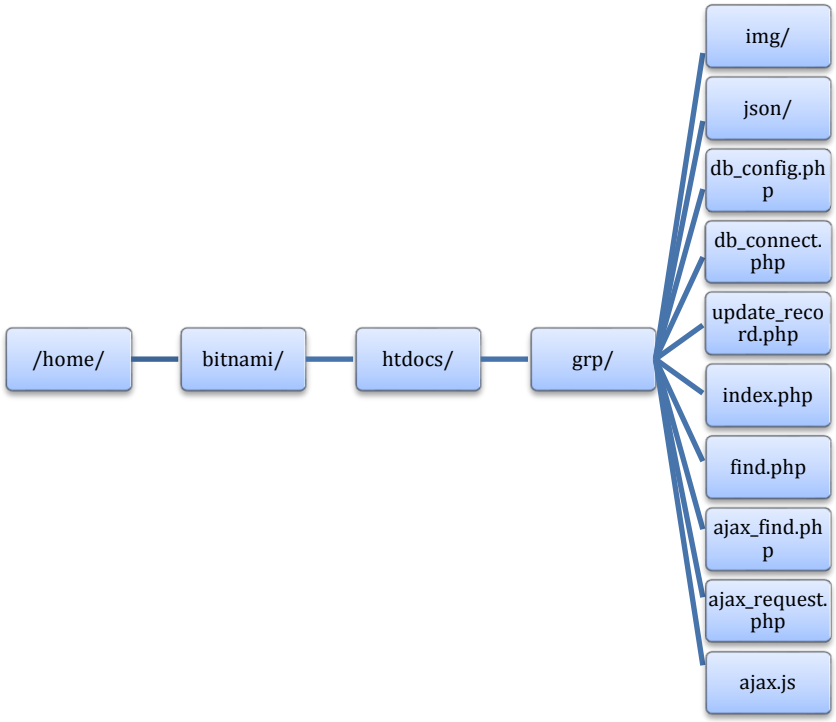


Figure 5.4: Server Directory

5.4.1 PHP SCRIPTS

All the core functions of our system involves sending data to and fetching data from the database. We plan to use the embedded SQL libraries of PHP to access the MySQL database in the server localhost. Therefore, we populated our server with PHP scripts designed to accomplish this.

db_config.php merely contains all the database connection variables to be used to connect to the MySQL database later on. Four variables are defined and they are as follows:

DB_USER: root
DB_PASSWORD: bitnami
DB_DATABASE: wireless_parking
DB_SERVER: localhost

db_connect.php is the main engine which does the work of establishing a connection with the MySQL database. Before doing so, it first imports the constant variables defined in db_config.php. Then, it uses the DB_USER and DB_PASSWORD to log in to MySQL and DB_DATABASE to connect to our

selected database. Finally, it returns the connection to whichever PHP scripts requesting for it.

update_record.php is the server backend of the sensor app. It receives two HTTP parameters, Sensor ID and Status, sent from the sensor via POST method. These two parameters will be submitted as part of the MySQL query to update the database. The MySQL command we use for this is UPDATE and with the WHERE clause we selectively update the record which meets the condition. In our case, we select the record which has the same Sensor ID as the Sensor ID parameter and update its Status field with the Status parameter sent over. The PHP script will check whether the MySQL operation succeeded or failed and generate a response message. The response message will be converted to JSON before it is echoed back to the sensor app through the web.

index.php defines the main map layout requested by the WebView in the Map interface. It contains a method call to the **ajax_request.php** through the **ajax.js** engine.

find.php defines the map layout requested by the WebView in the Find and Where interfaces. It contains a method call to the **ajax_find.php** through the **ajax.js** engine.

5.4.2 OPEN SOURCE LIBRARIES

Our server-side map rendering implementation uses the following open source modules:

AJAX JSMX LIBRARY

JSMX is an Ultra Lightweight - Language Agnostic - Ajax Framework. It is by far the easiest way to integrate Ajax into any Web Application. What separates JSMX from most other Ajax Frameworks is that the JSMX API runs entirely on the client and has no Server Side Components to install. Given this fact plus the fact that you can pass back JavaScript, XML, JSON, or WDDX makes JSMX a truly Universal Ajax API.

The beauty of this API is its simplicity and its straight forward syntax. It consists of just one file (**ajax.js**) which marshals requests between the client and the server via the http() function.

The diagram of the JSMX API is shown below. We place the ajax.js file in our display page. We then write two functions within JavaScript for each call to the server, a **request function** and a **callback function**. The request function makes either a "GET" or "POST" to the server by calling the http() function.

The server fires off the request and returns JavaScript, XML, JSON, or WDDX to ajax.js which in turn converts the response into a valid JavaScript Object and returns it to the callback function in your display template.

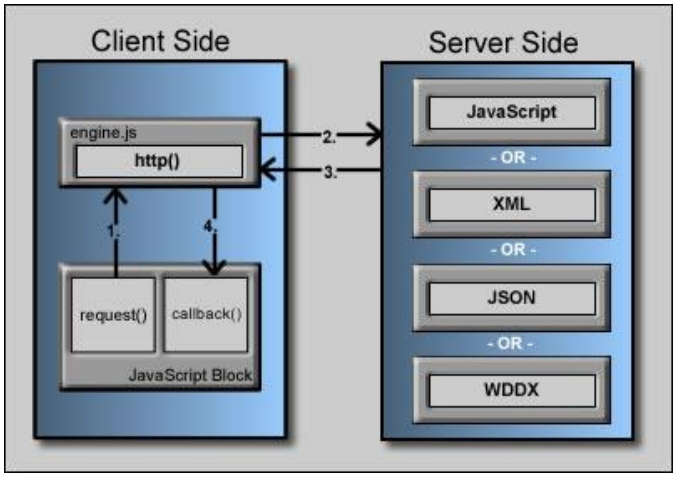


Figure 5.5: Client Server Relation

JSON-PHP LIBRARY

This package provides a simple encoder and decoder for JSON notation. It is intended for use with client-side Javascript applications that make use of HTTPRequest to perform server communication functions - data can be encoded into JSON notation for use in a client-side javascript, or decoded from incoming Javascript requests. JSON format is native to Javascript, and can be directly evaluated with no further parsing overhead.

6 SYSTEM TESTING

6.1 INTEGRATION TESTING

Integration Testing involves integrating all the system components together before conducting tests on the system as a whole. In an integration testing, we feed input data to the system via one end and expect the correct output from another end. With this we can inspect the data flow within the system and ensure that the communications between components were error-free. We carried out our integration testing by setting up a mini prototype model of the car park environment. Then, we used our sensor application to detect the vehicles on the mock model. On the other end, we examined the results displayed on the user application. If the results were as expected, this means that the database status has been updated successfully by the sensor and the user application also successfully retrieves the correct data from the database.

With this, we can safely verify that the data flow within the system is successful.

6.2 ACCEPTANCE TESTING

Acceptance Testing was conducted with our clients being the examiner. In our case, our client was our supervisor. After our system has passed the integration testing, we proceed to acceptance testing by our supervisor. Only user application is being tested in the Acceptance test because it is the only application that will be released to the clients.

The clients test our user application by inputting request to the application and the type of output determines whether the acceptance test was passed. Apart from our supervisor, a few of our course mates also conducted acceptance testing on our system.

7 SOFTWARE DEVELOPMENT TOOLS

7.1 GITHUB

GitHub is a web-based hosting service for software development projects that uses the Git revision control system. GitHub offers free accounts for open source projects. All members of the team are obliged to register a user account from GitHub and our repository master created a repository that we all can contribute in. Our online repository is named “wireless-parking-sys” and can be visited at https://github.com/jakebooyah/wireless-parking-sys.

Within the centralized repository, we host our project development directory, which includes source code and documentation materials. A centralized repository ensures that each of us always has reference to the latest version of the project files and everyone can make changes to the same copy of files. This avoids unnecessary branching from the master branch. Each group member has a local copy of repository in their machine and has full control over it. Changes we make to our local repositories can be pushed to the centralized repository so that everyone in the group can see it. For instances when more than one developers work on the same file simultaneously, GitHub has the mechanism to record and merge all the changes made.

7.2 ADT BUNDLE

As new Android developers, we downloaded the ADT Bundle to start developing our apps. It includes the essential Android SDK components and a

version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline our Android app development.

7.2.1 ANDROID SDK

The Android SDK provides the API libraries and developer tools necessary to build, test and debug apps for Android. The Android SDK Tools package that comes with the ADT Bundle is not the complete SDK environment but includes only core SDK tools. By using the Android SDK Manager, we downloaded a few other packages that are required for our app development, including Android Support Library, Google Play services and Intel x86 Emulator Accelerator (HAXM).

7.2.2 ECLIPSE IDE WITH BUILT-IN ADT

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give us developers a powerful, integrated environment in which to build Android applications.

ADT extends the capabilities of Eclipse to let us quickly set up new Android projects, create an application UI and debug our applications using the Android SDK tools. We really appreciate the guided project setup provided by ADT, as well as tools integration, custom XML editors and debug output pane. Without all these tools, we will not be able to initiate a smooth start in our Android development and complete it in a short time frame.

8 DISCUSSION

8.1 SUMMARY OF PROJECT MILESTONE ARCHIEVED

Basically, our group manage to create a parking system application which operates using three basic components:

- A sensor
- An online database
- An end user application

These three components are inter-related with each other. These three components were developed based on the requirement that our group set in the early stage of development. Since our group used the agile development

methodology in developing this application, the requirements change from time to time depending on the abilities of our group members and the time period set out in developing it.

For the sensor device, we managed to meet the requirements that our group discuss in the early stages. We used the camera from an android phone as a sensor, through a custom-built application which can be used on any android mobile device. This is a more eco-friendly approach that cuts out costs for the small scale of our project. By using phone's camera as a sensor, we were able to easily show a demonstration on a model map that we drew up combined with Lego cars as sample models. A program worked by detecting the changes in pixels on our model map, which then triggers a signal to be sent that changes the status of the related parking spot, thus determining the availability of the car park.

An online database was also created. This fulfilled the purpose of storing the parking statuses of the parking locations, and using it as the medium of updating the end user's Android phone. The server was programmed using SQL language and PHP scripts as the point of interaction. The sensor updates the database every five seconds as set in the camera application to capture and process image every five seconds. This data is then shared amongst all the users, in a web view within the Android mobile application as a map. The map highlights if that particular parking lot was taken and if it is not taken, the parking lot will not be highlighted and coloured based on the parking types which are red and yellow.

The mobile application implemented entailed one large map area – implemented using web view –with two building locations and two types of parking – either red or yellow. This can then be queried to show the available parking locations based on the criteria of the type of parking and proximity to convenient buildings.

Besides that, this application also allows the users to save the location where they park their car. This is to help the user to remember where they park their can just in case they forgot where they park. They just need to click save when they park their car at a particular parking lot. Since this system is not connected with GPS service, the users are required to input their current location to make them easier to find a car park and go to their car park position.

In terms of the user interface, a minimalistic and up-to-date approach was taken. These included commonly used icons which relate to the page being identified, a navigation drawer, toasts for guidance and content-focused views.

Lastly, with the aim of making the application more user friendly, we included a user manual in our application. This to ensure user that our end-user can easily use the application without much difficulty and on the least possible learning curve, with toasts being provided at each start up on various key concept within the app. This can be later switched off once the user can user the application comfortably.

8.2 REFLECTIVE COMMENTS ON THE PROJECT

Our group faced a mini crisis during the initial period of the current project where one of our members decided to drop out from the school. Since every student is already assigned with a group, it was impossible to get a replacement for the departed member. The crisis left us no choice but to move on with completing the project. Despite the fact that our group consists of only four members, the project is generally a great success.

The project topic itself came from a few intensive brain-storming sessions with our supervisor and eventually we decided to take up the challenge of creating a wireless parking system for the university as car parking is a serious issue in our campus. It was a brilliant idea to implement the system by using recycled android phones acting as sensors to our system but it was a challenge to the group as all of us have zero experience dealing with development in Android operating system. The problem was overcome by patience and dedication in self-educating ourselves in this particular field.

We started off the project by figuring out the implementation of the sensor that was meant to detect a vehicle. The image processing knowledge and Java coding experience from previous semester came in handy as Android source code shares a great similarity in language with Java and most phones are equipped with a camera. As the developers of Android discourage writing custom applications manipulating the camera module and encourage using the existing camera application in the operating system itself to handle image capturing, this posed a problem to us as our sensor has to be automated while implementing the encouraged method requires manual user intervention to complete the task. Despite the custom camera applications, including their documentation, are scarcely available on the web, we managed to create an application that captures images periodically and implemented image processing algorithms to the images. With those, we succeeded in creating a working application that turns an Android phone into a sensor for the system.

In the meantime, our group members were also researching for the methods to create an accessible online database and connect Android with it. We discovered that Amazon Web Services offers Amazon Elastic Compute Cloud

(EC2) which fulfils our needs and would not cost us to acquire the service for the time being. With some efforts, we acquired a server and managed to set up a MySQL database in it. Since then, our development phase has shifted to accomplishing the goal of connecting our sensor application to the database. At first we were thinking of connecting Android directly to MySQL or by using the Java Database Connectivitiy (JDBC) API. However, some research and example user cases in the forums proved to us that this is not feasible and highly error-prone. We also realized that sending bare MySQL statements over HTTP exposes security vulnerability in our system that can be exploited by SQL injection attack. In the end we resolved to a most commonly used and reliable method, which is by creating a web service on the Android platform, sending HTTP requests to the server-side PHP/MySQL platform. Based on AndroidHive's open source guide, we managed to set up working client-server architecture between our sensor application and the MySQL database.

Once we had successfully implemented the sensors and the server of our system, our next aim was to tackle the Android application to be made for the users of our system. We managed to get the user interface done as proposed in the interim report but according to the Android design guideline it was redundant to have a main menu page and an interactive navigation drawer that do the same thing. Therefore, we decided to remove the main menu and have the navigation drawer instead. The challenging part of creating the user application was the map with parking bays details and status. As we had no experience in creating user interface on Android operating system, this posed a difficult issue to us to actually create a map interface which constantly updates itself. Luckily, we thought of an idea to implement the map in a web view. Since WebView in Android library has panning and zooming features that a map requires, this solved our problem and JavaScript further simplified our job by providing the mechanism to manipulate the map display in a web layout. The extra features of our application were implemented without much difficulty as we've had the map implementation figured out. For the Find feature, we decided to send the user queries to the server using HTTP GET method, which generates different URLs for different variation of user queries. With this, each user can have a separate instance of the map view corresponding to their requests.

At the end of the day, we have a full working system with real time updates and the pace of the project was well distributed with the help of applying agile methodology in our project management. Sprints were carried out weekly and the task distribution was well planned and carried out (refer to Minutes in appendix for timeline). The project was broken down to smaller modules and tackled with focusing on getting small modules working in prototypes. This helped us by keeping the focus on issues with priorities and handling them step-by-step systematically towards the completion of the whole system.

9 BIBLIOGRAPHY

MindTouch Core. (n.d.). *How to connect to your Amazon Instance?* Retrieved from Bitnami: Amazon Cloud Images Quick Guide: http://wiki.bitnami.com/Amazon_cloud/how_to_connect_to_your_amazon_instance#How_to_connect_to_my_server

RaviTamada. (2012, May 2). *How to connect Android with PHP, MySQL*. Retrieved from AndroidHive: http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/

AWS. (n.d.). *What Is Amazon EC2?* Retrieved from AWS Documentation: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

Lalabird.com. (n.d.). *About JSMX*. Retrieved from JSMX: It's AJAX Made Simple: http://www.lalabird.com

The PHP Group. (n.d.). *Proposal for "Services_JSON"*. Retrieved from PEAR: http://pear.php.net/pepr/pepr-proposal-show.php?id=198

Simple AJAX. (n.d.). Retrieved from Simple Tutorials: http://simpletutorials.com/?path=tutorials/javascript/simple_ajax

Get the Android SDK. (n.d.). Retrieved from Android Developers: http://developer.android.com/sdk/index.html

ADT Plugin. (n.d.). Retrieved from Android Developers: http://developer.android.com/tools/sdk/eclipse-adt.html

10 APPENDIX

10.1 TEST CASES

Test Case: 1.1	Test Case Name: Detect vehicle
System: Sensor	Subsystem: Camera
Short Description: Camera should be able to detect presence of vehicle by comparing images captured.	

Pre-conditions:

The camera is working properly.

The camera has been set up at the parking lot.

The initial availability of the parking lot is empty.

Step	Action	Expected System Response	Pass/Fail
1	Activate Sensor app	An instance of camera obtained.	Pass
		SurfaceView created.	Pass
		SurfaceHolder passed to preview display.	Pass
		Preview surface started and keeps updating.	Pass
		Timer instance created.	Pass
		Recurring TimerTask scheduled by Timer to be executed every 7000 millisecs. (7 secs)	Pass
		TimerTask includes image capture and progress log update.	Pass
		Image data supplied by PictureCallback	Pass

		interface from photo capture.	
		Image stored as Bitmap in the RAM.	Pass
		Successive images are compared for their difference in pixels.	Pass
		CAR ABSENT: Sensor decides vehicle absent if pixel difference is lower than a predefined threshold.	Pass
		CAR ABSENT: Sensor decides vehicle present if pixel difference exceeds a predefined threshold.	Pass
		NO INTERNET: Error message to notify user.	Pass
		CAR ABSENT: HTTP POST request submitted to server-side PHP script to update MySQL database.	Pass
		CAR PRESENT: Sensor decides vehicle present if pixel difference is lower than a predefined threshold.	Pass
		CAR PRESENT: Sensor decides vehicle absent if pixel difference exceeds a predefined threshold.	Pass
		NO INTERNET: Error message to notify user.	Pass
		CAR PRESENT: HTTP POST request submitted to server-side PHP script to update MySQL database.	Pass
2	Before parking	Sensor will not capture significant changes between successive image captures.	Pass

	a vehicle on the parking bay	CarPositive state set to FALSE.	Pass
3	After parking a vehicle on the parking bay	Sensor captures higher-than-threshold difference between the latest image and the image immediately before.	Pass
		Parking bay status set to 1.	Pass
		CarPositive state set to TRUE.	Pass
		HTTP POST request submitted to server-side PHP script to update MySQL database.	Pass
		HTTP response from server received by the sensor.	Pass
4	Leave the vehicle on the parking bay	Sensor will not capture significant changes between successive image captures.	Pass
		Parking bay status remains at 1.	Pass
		CarPositive state remains of TRUE.	Pass
5	Empty the parking	Sensor captures higher-than-threshold difference between the latest image and the image immediately before.	Pass

	bay	Parking bay status set to 0.	Pass
		CarPositive state set to FALSE.	Pass
		HTTP POST request submitted to server-side PHP script to update MySQL database.	Pass
		HTTP response from server received by the sensor.	Pass

Test Case: 1.2	Test Case Name: Update database
System: Sensor	Subsystem: Server
Short Description: Update the database on the parking bay status.	

Pre-conditions:

The camera system is turned on and working properly at parking slot.

There is a change in availability in car park slot

Step	Action	Expected System Response	Pass/Fail
1	Connect via SSH to server	SSH connection with server established.	Pass
2	Execute PHP script (db_config.php)	Database connection variables defined.	Pass
		Username correctly defined.	Pass
		Password correctly defined.	Pass
		Database name correctly defined.	Pass
		Database server correctly defined.	Pass
3	Execute PHP script (db_connect.php)	Connection with MySQL database established.	Pass
		Wireless_parking database selected.	Pass
4	Execute PHP script (update_r	Connection with database established.	Pass
		MySQL query generated.	Pass

	ecord.php)	Update operation successfully performed.	Pass
		Response fields generated.	Pass
		Response output printed.	Pass
5	Combine sensor with server-side implement ation	Sensor_ID and Status parameters received through HTTP connection.	Pass
		“Status” field of the database record pointed to by the Sensor_ID parameter successfully updated.	Pass
		Response field generated.	Pass
		Response output printed.	Pass
		Server response successfully received by the sensor.	Pass
6	Before parking a vehicle on the parking bay	“Status” field of the specified Sensor_ID record is 0.	Pass
7	After parking a vehicle on the parking bay	“Status” field of the specified Sensor_ID record is set to 1.	Pass
8	Leave the vehicle on the parking bay	No changes made to the “Status” field of the specified Sensor_ID record.	Pass

9	Empty the parking bay	“Status” field of the specified Sensor_ID record is set to 0.	Pass

Test Case: 1.3	Test Case Name: Start user app
System: User Application	Subsystem: User Interface
Short Description: Turn on the Android user application	

Pre-conditions:

The server and camera system is functioning properly.

Step	Action	Expected System Response	Pass/Fail
1	Touch the app icon on menu	Application successfully launched.	Pass
		NO INTERNET: Error message to prompt user to check for network.	Pass
2	Swipe from the left edge of the screen	Navigation drawer expanded.	Pass
		Current action bar title replaced with app name.	Pass
		Map navigation option displayed.	Pass
		Find navigation option displayed.	Pass
		Where navigation option displayed.	Pass
		Settings navigation option displayed.	Pass
3	Touch the app icon on the action bar	Navigation drawer expanded.	Pass
		Current action bar title replaced with app name.	Pass
		Map navigation option displayed.	Pass

		Find navigation option displayed.	Pass
		Where navigation option displayed.	Pass
		Settings navigation option displayed.	Pass
4	Touch the content outside the navigation drawer	Navigation drawer dismissed.	Pass
5	Swipe to the left anywhere on the screen	Navigation drawer dismissed.	Pass
6	Touch the app icon/title in the action bar	Navigation drawer dismissed.	Pass
7	Press Back	Navigation drawer dismissed.	Pass
8	Touch Map option	Map user interface displayed.	Pass
9	Touch Find option	Find user interface displayed.	Pass
10	Touch Where option	Where user interface displayed.	Pass
11	Touch Settings option	Settings user interface displayed.	Pass
12	Press Back	Application successfully terminated.	Pass

Test Case: 1.4	Test Case Name: Start Map
System: User Application	Subsystem: Map
Short Description: Obtain car park availability information	

Pre-conditions:

The server and camera system is functioning properly.

Step	Action	Expected System Response	Pass/Fail
1	Turn on Map interface	Map layout displayed.	Pass
		Red parking zone displayed.	Pass
		Yellow parking zone displayed.	Pass
2	Pinch open on Map view	Map layout zoomed in.	Pass
3	Pinch close on Map view	Map layout zoomed out.	Pass
4	Swipe/drag across Map view	Map layout scrolled.	Pass
5	Before parking a vehicle on the red parking bay	Red parking bay appears red.	Pass
6	After parking a vehicle on the red parking bay	Red parking bay appears dark grey.	Pass
7	Leave the vehicle on the	Red parking bay appears dark grey.	Pass

	parking bay		
8	Empty the parking bay	Red parking bay appears red.	Pass
9	Before parking a vehicle on the yellow parking bay	Yellow parking bay appears yellow.	Pass
10	After parking a vehicle on the yellow parking bay	Yellow parking bay appears dark grey.	Pass
11	Leave the vehicle on the parking bay	Yellow parking bay appears dark grey.	Pass
12	Empty the parking bay	Yellow parking bay appears yellow.	Pass

Test Case: 1.5	Test Case Name: Find parking
System: User Application	Subsystem: Find
Short Description: Find empty parking slots according to user preferences	

Pre-conditions:

The server and camera system is functioning properly.

Step	Action	Expected System Response	Pass/Fail
1	Turn on Find interface	Find layout displayed.	Pass
		Map layout displayed.	Pass
		User input pane displayed.	Pass
2	Pinch open on Map view	Map layout zoomed in.	Pass
3	Pinch close on Map view	Map layout zoomed out.	Pass
4	Swipe/drag across Map view	Map layout scrolled.	Pass
5	Check red parking checkbox	Vacant red parking bays appear blue.	Pass
6	Check yellow parking checkbox	Vacant yellow parking bays appear blue.	Pass
7	Check red and yellow parking	Vacant parking bays, regardless of	Pass

	checkboxes	colour, appear blue.	
8	Select a “nearest to” building	RED CHECKED: The nearest vacant red parking bays to the selected building appear blue.	Pass
		YELLOW CHECKED: The nearest vacant yellow parking bays to the selected building appear blue.	Pass
		RED & YELLOW CHECKED: The nearest vacant red and yellow parking bays to the selected building appear blue.	Pass

Test Case: 1.6	Test Case Name: Save location
System: User Application	Subsystem: Where
Short Description: Saved parking location	

Pre-conditions:

The server and camera system is functioning properly.

Step	Action	Expected System Response	Pass/Fail
1	Go to Where interface	Where layout displayed.	Pass
		Map layout displayed.	Pass
		User input pane displayed.	Pass
2	Pinch open on Map view	Map layout zoomed in.	Pass
3	Pinch close on Map view	Map layout zoomed out.	Pass
4	Swipe/drag across Map view	Map layout scrolled.	Pass
5	Select a “you parked at” parking bay	Selected parking bay ID shown on the “your saved location” panel.	Pass
		Selected parking bay appears blue.	Pass
		Selected parking bay ID saved in Shared Preferences.	Pass
6	Navigate to other	Saved parking bay ID remains on	Pass

	interface then switch back to Where	the “your saved location” panel.	
		Saved parking bay appears blue.	Pass
		Saved parking bay data persists.	Pass
7	Restart app and go to Where interface	Saved parking bay ID remains on the “your saved location” panel.	Pass
		Saved parking bay appears blue.	Pass
		Saved parking bay data persists.	Pass
8	Select a new “you parked at” parking bay	Newly selected parking bay ID shown on the “your saved location” panel.	Pass
		Newly selected parking bay appears blue.	Pass
		Newly selected parking bay ID overwrites the old parking bay ID in Shared Preferences.	Pass
9	Navigate to other interface then switch back to Where	Newly saved parking bay ID remains on the “your saved location panel”.	Pass
		Newly saved parking bay appears blue.	Pass
		Newly saved parking bay data persists.	Pass
10	Restart app and go to Where interface	Newly saved parking bay ID remains on the “your saved location panel”.	Pass
		Newly saved parking bay appears	Pass

		blue.	
		Newly saved parking bay data persists.	Pass

10.2 MINUTES OF MEETINGS

Meeting: Informal 1
Date: 8/10/2013
Time: 1100 - 1130
Venue: Library Meeting Room 2 (GA11A)

Attendance:
Zhi En
Michael
Wei Qi
Miow Fang
Syamil

Discussed:

-Roles
1) Zhi En - Coder, Open Day Producer, Leader
2) Mike - Editor, Open Day Presenter
3) Syamil - Editor, Artistic Director
4) Wei Qi - Coder, Artistic Director
5) Miow Fang - Coder, Repository Master

6) Everyone - Software Architecture, Quality Tester

-Meeting Roles
Rotate each week by sequence "Zhi En, Michael, Syamil, Wei Qi, Miow Fang"
Chair:Secretary

-Meeting Slots
1) Tuesday: 1100 - 1300
2) Wednesday: 1200 onwards
3) Friday: 0900 - 1100
4) Weekends: Maybe?

-Set an informal and formal meet each week

To Do:

-Discuss topic with supervisor
-Set regular formal meeting day
-Ask supervisor advices
-Inform roles

Meeting: Informal 2
Date: 8/10/2013
Time: 1400 - 1430
Venue: Selvaraj's Office

Attendance:
Zhi En
Michael
Wei Qi
Miow Fang
Syamil

Discussed:

-Informed roles and progress
-Asked for supervisor advices
-Tutor advises to have a selection of ideas, to choose from and expound fully before making a choice

To Do:

-Find ideas on topics
-Discuss topics

Meeting: Formal 1
Date: 11/10/2013
Time: 1200 - 1300
Venue: Selvaraj's Office

Attendance:
Zhi En
Michael
Wei Qi
Miow Fang
Syamil

Discussed:

-Ideas on topic
1) Timetabling app (to-do list + reminder)
2) An electronic approach to replace current university canteen coupon system
3) Car locator app
4) Web browser with built-in porn filter
5) Property Management for TTS
6) Money Planner
7) Cinema Access Card system instead of conventional ticketing
8) Android keyboard input

-Practicality of each topic

-Finalized topics
1) Timetabling app (to-do list + reminder)
2) An electronic approach to replace current university canteen coupon system
3) Property Management for TTS
4) Money Planner
5) Android keyboard input

To Do:

-Find more ideas on topics
-Brainstorm on finalized topics in details

Meeting: Informal 3
Date: 16/10/2013
Time: 1600 - 1800
Venue: GA11a

Attendance:
Zhi En
Michael
Wei Qi
Miow Fang
Syamil

Discussed:

-Brainstorming on last week's finalized topic

1) Timetabling app (to-do list + reminder)

Who
- University of Nottingham student

Why
- Link Nottingham webpage to check timetable
- DB (online / download related module package)
- Ease timetabling scheduling for UNMC students
- Module based data schedule
- Including full year academic timetable and bus schedule

Constraints
- How to get timetable from university
- iOS community?

How?
- Android (JAVA)
- Online DB (SQL)
- Timetable (Gabriel Ittner)

2) An electronic approach to replace current university canteen coupon system

Who?
- UNMC people

Why?
- Coupon system is not eco-friendly
- Coupon not lasting
- Convenience if integrated with ID card

Constraint
- Requires a lot of hardware
- Coding
- External policy problem

How?
- Use student card (need to understand how student ID works)
- dummy DB
- C programming language/JAVA

3) Property Management for TTS

Who?
- UNMC students
- Landlords in TTS

Why?
- Currently there are no available management system
- Difficult to identify and advertise house
- Visualisation of the condition of the place
- Easier to allocate the available houses or room to rent

Constraint
- Will require training
- Seasonal usage
- Many instances and difficult to program
- Difficult to promote

How?
- XML, PHP, HTML
- Framework
- DB (SQL)

4) Car park Sensor App

Who?
- UNMC students and staff

Why?
- Current car park system is not user friendly
- let all the staff and student easier to find parking slot
- can find parking a lot faster and won't be late for class
- reminding student and staff about bus as well

Constraint
- Sensor (hardware)
- Weather effect on the functionality of the sensor
- requires data plan or internet
- Dangerous for the driver to use especially when they are driving
- lagging in information/not accurate

How?
- Android (java)
- have to research on hwo the sensor works
- The university parking slot map
- Online database

5) Android keyboard input

Who?
- Android user

Why?
- Current keyboard is hard to use
- change the way how people interact with their phone
- New invention

Constraints
- Difficult current code
- Harder to use using one hand
- will the community accept?
- Difficult to implement in tablet

How?
- Android (JAVA)
- Local DB
- Requires internet to update library of words

To Do:

- Present all the result of this brainstorming session to Mr Selvaraj during next Formal meeting
- Brainstorm on finalized topics in more details

Meeting: Formal 2
Date: 18/10/2013
Time: 1200 - 1300
Venue: Selvaraj's Office

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Deep discussion about what have been decided in brainstorming session
1) Timetabling app (to-do list + reminder)
2) An electronic approach to replace current university canteen coupon system
3) Property Management for TTS
4) Carpark Sensor App
5) Android keyboard input

-Narrow down the project discussion into 3 options
1) Property Management for TTS
2) Carpark sensor app
3) Android keyboard input

To Do:

- Need to rearrange the structure of the group since one of our group mate decided to leave the university
- Need to do research on those three finalized idea.

Meeting: Informal 4
Date: 22/10/2013
Time: 1100 - 1300
Venue: The Core Room 3A (HA18H)

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Roles Update

1) Zhi En - Coder, Open Day Producer, Leader
2) Mike - Coder, Editor, Open Day Presenter
3) Syamil - Editor, Artistic Director
4) Wei Qi - Coder, Artistic Director, Repository Master
5) Everyone - Software Architecture, Quality Tester

-Meeting Roles update

Rotate each week by sequence "Zhi En, Michael, Syamil, Wei Qi"
Chair:Secretary

-Pointed out importance of Asana (Task Scheduler)

-Added new topic to pool of ideas

Warehouse Transport System

Who?
- Warehouse manager

Why?
- Automated system
- Human Error could be avoided
- Time saving
- Cost efficeint in long term
- Efficient work

Constraint
- Small scaled prototype
- Requires Hardware

How?
- BrickPi/NXT Brick
- Sensors and motors from Lego Mindstorm
- C programing/NXT-G
- Request NXT set from school

-Brainstormed and evaluated each topic
-Group decided to go for Warehouse Transport System
-Talked to supervisor about decided topic

To Do:

-Read up about references on chosen topic
    http://www.dexterindustries.com/BrickPi/
    http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT_2.0
    http://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547

-Confirm topic

Meeting: Informal 5
Date: 24/10/2013
Time: 1400 - 1600
Venue: BB81

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Switching to Car Park System due to numerous difficulties in implementing transport system.
    eg. Hardware issues, electronics circuits

-Consulted Mr. Ben on the hardwares and programing languages required to implement Car Park System.

-Decided to use old andriod phones as sensor

-2 applications needed:
    1) For the users that display the map and available parking spaces
    2) For the sensor phone to communicate with the database

To Do:

-Update meeting slots
-Consult Mr. Selva about changes
-Final confirmation on topics
-Find out ways to use phone camera to detect vehicles

Meeting: Formal 3
Date: 18/10/2013
Time: 1200 - 1300
Venue: Selvaraj's Office

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Confirmed Car Park System topic

-Consulted Mr. Selva on report formats and the time planning

-Consulted Mr. Selva on software engineering methodology

To Do:

-Update meeting slots

-Find out ways to use phone camera to detect vehicles

-Project Description

-Read up on Software Engineering methods

Meeting: Informal 6
Date: 30/10/2013
Time: 1100 - 1300
Venue: Wei Qi's house

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Decided on the project title: Wireless Parking Sensor System

-Drafted the project description. It consists of 3 sections:
1) Problem description
2) Objectives
3) Solution

-Updated our available meeting time slots:
1) Monday: 0900 - 1100
2) Tuesday: 1100 - 1300
3) Wednesday: Available entire day
4) Thursday: 1300 - 1400
5) Friday: 0900 - 1300

To Do:

-Find out ways to use phone camera to detect vehicles
-Prepare and complete the project description for submission
-Decide on Software Engineering methodology to be employed
-Set out a time plan for the project

Meeting: Formal 4
Date: 31/10/2013
Time: 1415 - 1500
Venue: Selvaraj's Office

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Revised format of the project description:
    1) Aim
    2) Objectives
    3) Background and Motivation
    4) Project Plan

To Do:

-Make changes to the project description according to the revised format.
-Make a Gantt chart of our project planning.
-Print out a copy of submission deadlines for our reference.
-Decide on Software Engineering methodology to be employed.
-Find out ways to use phone camera to detect vehicles.

Meeting: Informal 7
Date: 7/11/2013
Time: 1700 - 1900
Venue: Zhi En's place

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Time plan for interim report completion, and assigning broken down tasks to group members:
-Rough sketch of Gantt chart

To Do:

-Complete Interim documentation (Different sections for each member)
-Create designs for the solution to be implemented

Meeting: Informal 8
Date: 25/11/2013
Time: 1100 - 1300
Venue: Library Meeting Room

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-User's stories
-Requirement Specifications

To do:

-Research on user interface and data model
-Discuss user interface and data model

Meeting: Informal 9
Date: 28/11/2013
Time: 1100 - 1300
Venue: BB81

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-User Interface in each and every cases
-Data flow model
-Model View Controller
-Use Cases
-Gantt Chart review

To do:

-Study use cases and discuss coming up meeting
-First prototype of user interface
-complete data modelling

Meeting: Informal 10
Date: 4/12/2013
Time: 1100 - 1300
Venue: Wei Qi's place

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Review on GUI design and use cases diagram
-Discuss on the key implementation records and problems encountered so far
-Draft out full-year project time plan (Gantt chart)
-Divide tasks among members to complete the Interim Report

To do:

-Syamil: Discussion of problems + Time plan + Gantt chart
-Michael: Key implementation record
-Wei Qi: GUI and algorithm explanation
-Zhi En: Development methodology + DFD & use case explanation

Meeting: Informal 11
Date: 14/2/2014
Time: 1600 - 1800
Venue: Wei Qi’s place

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Progress over the holiday
-Divided coding jobs
-Jake and Zhi En work on taking pic and image processing algorithm
-Syamil and Mike work on online server

To do:

-Study and research on individual's field

Meeting: Formal 5
Date: 20/2/2014
Time: 1600 - 1700
Venue: Selvaraj's Office

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

-Updated Selvaraj on our progress over the holidays.
-Asked for comments on our Interim Report.
-Project's technical limitations to be solved:
1) Camera cannot take picture clearly at night.
2) Camera might falsely recognise non-vehicle objects (e.g. human, animals) as vehicles.
3) Difficulty in collecting actual map of the campus and integrating it into our android app.

-Suggestions by Selvaraj:
1) Limit the operating hours of the system to office hours only.
2) Camera is constantly updating its status within every 5s interval.
3) Creating a parking lot prototype with simplified model of the campus map.

To do:

-Produce the first prototypes:
i) Camera module – Wei Qi & Zhi
ii) Server module - Mike & Syamil
-Implementation of the parking lot simulation model.

Meeting: Informal 12
Date: 27/02/2014
Time: 1600-1700
Venue: The Core

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

- Review and checking the milestone that has been release on the previous two weeks.
- Discuss the Progression of our group project until this week
- Divide new milestone for each group member to be completed.
- Discuss the layout of the map to be used.

To Do:

Continue doing the first prototypes:
1) Improving the camera module and algorithm - Jake
2) Making database - Zhi & Mike
3) Design the map - Syamil

Meeting: Formal 6
Date: 6/03/2014
Time: 1600-1700
Venue: Selvaraj's Office

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

- Showing the progression of our group project to Mr Selvaraj
- He gave some advice about our project:
  - Image processing interrupts your time (care)
  - try to make many demonstrate
  - need to have slack time (backup plan)
  - If possible make counter showing how many slot is free

To Do:

Make a small informal meeting straight away after the formal meeting to discuss about our next milestone.
1) Combining the developed part from last week
2) Making the UI of the application
3) Refining the maps

Meeting: Informal 13
Date: 14/03/2014
Time: 1600-1800
Venue: Wei Qi's Place

Attendance:
Zhi En
Michael
Wei Qi

Discussed:

- Showing the progression of our individual assignments
- Evaluation of what has been completed:
  - Checking on design of android app
  - testing of car detection app and tuning
  - Discussion of possible implementation of map
- Confirm workings of the server and ability to edit from android application

To Do:

1) Combining the server editing code with camera detection code
2) Make the side menu for the UI of the application
3) Figure out how to implement maps

Meeting: Informal 14
Date: 20/03/2014
Time: 1600-1800
Venue: Wei Qi's Place

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

- Showing the progression of our individual assignments
- Evaluation of what has been completed:
  - Demonstration of semi-completed side menu of android app (lack of ActionListener)
  - Testing of car detection app and further tuning to improve accuracy
  - Discussion of possible implementation of car park simulation model
    - How to build the car park model?
    - How to obtain car models?
    - How to implement camera holder?

To Do:

1) Complete the side menu with fully functioning ActionListeners.
2) Figure out how to implement maps on android app.
3) obtain the car models for testing and demonstration purposes.
4) Print out car park layout and create the whole simulation model hardware.

Meeting: Informal 15
Date: 27/03/2014
Time: 1600-1800
Venue: Wei Qi's Place

Attendance:
Zhi En
Wei Qi
Syamil

Discussed:

- Discussion on the poster design
- Discussion on how Where and Find feature will be implemented
- Map design for demonstration model

To Do:

1) Complete presentable prototype with find feature
2) Start on final report
3) Finalise poster design

Meeting: Formal 7
Date: 01/04/2014
Time: 1300-1400
Venue: Selvaraj Office

Attendance:
Zhi En
Michael
Wei Qi
Syamil

Discussed:

- Showing the progression of our group project to Mr Selvaraj:
- A working map display prototype which could be updated when then sensor sensed a car on the parking lot
  - A working prototype model of sensor and car model
- Discussion of future sprints tasks
- Seek advice about the poster design due 9th of April

To Do:

1) Complete the working prototype model of sensor and car
2) Figure out the way to hold the sensor
3) Distribute tasks on final group report writing
4) Complete the Open Day poster

Meeting: Informal 16
Date: 03/04/2014
Time: 1600-1800
Venue: Wei Qi's Place

Attendance:
Zhi En
Wei Qi
Syamil
Michael

Discussed:

- Discussion on how Where feature should be implemented
- Distribute the tasks of the final group report writing
- Map design for demonstration model

To Do:

1) Complete Where feature
2) Start on the report writing parts being assigned
3) Add a parking bay selector for the sensor application

Meeting: Informal 17
Date: 10/04/2014
Time: 1600-1800
Venue: Wei Qi's Place

Attendance:
Zhi En
Wei Qi
Syamil
Michael

Discussed:

- Review on the final report writing progress
- Distribute the remaining tasks of the final group report writing
- Review on the completed user application

To Do:

1) Finalize the entire software, including the sensor app and the user app
2) Complete the final group report
3) Add a camera flashlight function to the sensor app

Meeting: Formal 8
Date: 17/04/2014
Time: 1600-1700
Venue: Selvaraj's Office

Attendance:
Zhi En
Wei Qi
Syamil
Michael

Discussed:

- Full demonstration of our software to Mr Selvaraj
- Discussion on any parts left out in the final group report
- Discussion on any improvement needed on top of the report
- Advice for Open Day and Presentation Day

To Do:

1) Finalize the entire software
2) Finalize the final group report
3) Get ready for Open Day